VŠB – Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Tvorba úloh v prostředí ROBOT C pro LEGO Mindstrorms Education NXT

Creating Tasks in Robot C for LEGO Mindstorms Education NXT

Student: Adam Machač
Vedoucí bakalářské práce: Ing. David Fojtík, Ph. D.

Ostrava 2013

VŠB - Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Zadání bakalářské práce

Student: **Adam Machač**

Studijní program: B2341 Strojírenství

Studijní obor: 3902R001 Aplikovaná informatika a řízení

Téma: Tvorba úloh v prostředí ROBOT C pro LEGO Mindstorms Education NXT
Creating Tasks in ROBOT C for LEGO Mindstorms Education NXT

Zásady pro vypracování:

1. Popište stavebnici LEGO Mindstorms Education NXT , zaměřte se na její vybavení včetně doplňkových senzorů a pohonů aktuálně dostupných na trhu.
2. Popište možnosti programovacích nástrojů LEGO Mindstorms Education NXT. Zaměřte na programovací prostředí ROBOT C, pro které zpracujte základní studijní příručku.
3. Pomocí stavebnice Mindstorms Education NXT a programového prostředí ROBOT C realizujte sadu demonstračních úloh a podrobně je popište.
4. Zhodnoťte dosažené výsledky a navrhněte směry dalšího řešení.

Seznam doporučené odborné literatury:

DALE YOCUM [online]. *NXT Tutorial.* Dale Yocum. 2007. dostupné z WWW: <URL: http://www.ortop.org/NXT_Tutorial/>.
EDUXE, *LEGO Educational Division v ČR*, aktualizace 2010, [online], [cit. 2010], dostupné z URL <http://www.eduxe.cz/>
HEROUT, P. *Učebnice jazyka C.* 1.díl 5.vyd. Praha : KOPP Praha, 2008. 280 str. - ISBN: 80-7232-351-2
HiTechnics, *HiTechnic Products*, aktualizace 2010, [online], [cit. 2010], dostupné z WWW < URL: http://www.hitechnic.com/>
Hurbain, P. E. Gasperi, M. *Extreme NXT: Extending the LEGO MINDSTORMS NXT to the Next Level (Technology in Action)*. 2009, ISBN: 978-1430224532
LEGO *LEGO® Education*, aktualizace 2010, [online], [cit. 2010], dostupné z WWW <URL: http://www.legoeducation.us/global.aspx>
PHExtreme NXT, *Philo's Home Page*, aktualizace 2010, [online], [cit. 2010], dostupné z WWW <URL: http://www.philohome.com/>
ROBOTC *Teaching ROBOTC for LEGO MINDSTORMS.* [online] dostupné z WWW: <URL: http://www.robotc.net/>.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Fojtík, Ph.D.**

Datum zadání: 14.12.2012
Datum odevzdání: 20.05.2013

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta strojní 3 OSTRAVA

Tůma

prof. Ing. Jiří Tůma, CSc.
vedoucí katedry

doc. Ing. Ivo Hlavatý, Ph.D.
děkan fakulty

Místopřísežné prohlášení studenta

Prohlašuji, že jsem celou bakalářskou práci včetně příloh vypracoval samostatně pod vedením vedoucího bakalářské práce a uvedl jsem všechny použité podklady literaturu.

V Ostravě ……. …….

Adam Machač

Prohlašuji, že

- jsem byl seznámen s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo.
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen „VŠB – TUO“) má právo nevýdělečně ke své vnitřní potřeb bakalářskou práci užít (§ 35 odst. 3).
- souhlasím s tím, že bakalářská práce bude v elektronické podobě uložena v Ústřední knihovně VŠB – TU k nahlédnutí a jeden výtisk bude uložen u vedoucího bakalářské práce. Souhlasím s tím, že údaje kvalifikační práci budou zveřejněny v informačním systému VŠB – TU.
- bylo sjednáno, že s VŠB – TU, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.
- bylo sjednáno, že užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB – TU, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB – TU na vytvoření díla vynaloženy (až do jejich skutečné výše).
- beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě: ……

……

podpis

Jméno a příjmení autora práce: Adam Machač

Adresa trvalého pobytu autora práce: Na Slezance 719/6, Vratimov

Anotace

Fojtík D Tvorba úloh v prostředí ROBOT C pro LEGO Mindstrorms

Ostrava: Katedra automatizační techniky a řízení, Fakulta strojní

VŠB – Technická univerzita Ostrava, 2013, s. Bakalářská práce,

Vypracoval: Machač A.

Práce se zabývá vytvořením jednoduché příručky pro programovací prostředí RobotC. Seznamuje se senzory základní sady LEGA Mindstorms. Popisuje také demonstrativní úlohy, kterými jsou hlídací robot – primitivní alarm, kalibrace senzorů, jízda po čáře pomocí dvou světelných senzorů, průjezd bludištěm a dálkově ovládaný robot, řízený zvukem. Tyto úlohy byly zpracovány a podrobně popsány, včetně použitých funkcí a využití, v následujících kapitolách.

Annotation

Fojtik D Production of tasks in the robotics programming langure ROBOT C for LEGO Mindstorm

Ostrava: Department of Control Systems and Intrumentation, Faculty of Mechanical Engineering

VŠB – Technical University of Ostrava, 2013, s. Project,

Author: Machač A.

Thesis deals with creation of simple manual for programming enviroment RobotC. It introduces sensors from the basic Lego Mindstorms set. It also describes demonstrative challenges which includes a guard robot - primitive alarm, sensor calibration, tracking the line using two light sensors, the maze and robot remotely-controlled by sound. These challenges were developed and detailly described including the functions and usage in the following chapters.

Klíčová slova

RobotC, LEGO Mindstorms, příručka, demonstrativní úlohy

Key words

RobotC, LEGO Mindstorms, manual, demonstrative role

Obsah

Seznam obrázků

Seznam zkratek

Tabulka 1 Seznam zkratek

NXT kostka	**Řídící jednotka robotické stavebnice**
RJ12	typ telefonního kabelu
USB	(**Universal Serial Bus**) je univerzální sériová sběrnice
AA	standardní tužkové baterie
dBA	akustický tlak
PID	proporcionální, integrační a derivační složka regulace
RCX	Předchůdce NXT řídící jednotky
int	(integer) formát čísla
Bluetooth	je v informatice proprietární otevřený standard pro bezdrátovou komunikaci propojující dvě a více elektronických zařízení
EV3	LEGO „nové generace“

Seznam použitých funkcí

Tabulka 2 Seznam použitých funkcí

SensorValue	**hodnota určitého senzoru**
motor	hodnota přednastaveného motoru
PlaySound(soundBeepBeep)	hraje jeden z přednastavených tónů
wait1Msec(….)	čekání po určitou dobu
nMotorPIDSpeedCtrl	zapne regulaci motorů
nMotorEncoder	vynuluje motory po uplynutí přibližně 90 otáček
nxtDisplayCenteredTextLine	vypisuje hodnoty senzorů na displej programovatelné kostky

Programujeme, dle zásad programování jazyka C. Používáme funkce a operátory jako např. main, if, else, &&, stejně jako v programování klasického jazyka C. Viz uvedený zdroj kniha jazyk C – Dalibor Kačmář.

Úvod

Cílem bakalářské práce je v prvé řadě seznámení s programovacím prostředím RobotC pro LEGO Mindstorms, jeho funkcemi, nastavením a nástroji. Zorientovat se v něm a naučit se v tomto prostředí pracovat. Programovací jazyk C je nejrozšířenějším jazykem, je velmi používaný, má spoustu modifikací a úprav uzpůsobených pro různé aplikace, programy a také pro různé prostředí. RobotC umožňuje uživatelům LEGO Mindstorms tvorbu složitějších úloh.

Dále také seznámení se senzory a motory, kabely a mozkem celé stavebnice, čímž je NXT kostka, které LEGO Mindstorms nabízí a se kterými je možno pracovat. Také seznámení s novinkami na trhu v oblasti LEGA Mindstorms.

Jedním z hlavních cílů bakalářské práce je utvoření jednoduché příručky prostředí RobotC. Popsání funkcí a orientace, pro rychlou práci s tímto prostředím a vytvoření sady demonstrativních úloh a jejich podrobné popsání.

1 LEGO Mindstorms

Stavebnice LEGO Mindstorms nejen hračka pro děti, určena výrobcem pro děti od osmi let. LEGO Mindstorms je jednoduchá programovatelná stavebnice. V této stavebnici si každý najde to své, je v podstatě neomezeně variabilní. Záleží na vynalézavosti a fantazii, ale také hravosti každého jednoho člověka, které kostky použije a jak je propojí, sestaví a zkombinuje logicky. Základním členem stavebnice a jakoby mozkem robota je chytrá programovatelná kostka NXT. K této kostce je možno připojit až tři servomotory, které jsou obsaženy v základní soupravě a až čtyři senzory, kterých je nespočetné množství. Na trhu si je možno vybrat mnoho senzorů, které jsou plně kompatibilní s programovatelnou kostkou NXT. Programovatelná kostka obsahuje také bluetooth zařízení. Toto zařízení zprostředkovává bezdrátový chod a pohyb a komunikaci. Ze stavebnice je možno sestavit různé roboty, jezdící, chodící, roboty reagující na barvu, zvuk, či pohyb. Stavebnice obsahuje svůj vlastní grafický programovací systém, který je velmi jednoduchý a plný obrázků. S tímto systémem mohou děti naučit svého roboty splňovat jednoduché, ale i složitější úlohy. Stavebnice LEGO Mindstorms, ale také podporuje několik programovacích jazyků. To je zajímavá skutečnost nejen pro zkušenější malé programátory, ale také pro dospělé a hlavně pro studenty a učitele. První typ stavebnice vydalo LEGO ve spolupráci se laboratoří MIT v roce 1998 pod názvem Robotics Invention System. Další verze vyšla v roce 2006 a to už pod názvem LEGO Mindstorms NXT a hlavním rozdílem je možnost připojení přes bluetooth a poté verze z roku 2009 Lego Mindtorms NXT 2.0, kde se již objevil nový barevný senzor. Poslední, nejnovější verze vyšla roku 2011. Verze, na které budou zpracovány úlohy této práce.

1.1 Programovatelná kostka

Hlavním a základním prvkem stavebnice je programovatelná kostka NXT. Kostka obsahuje 32 – bitový mikroprocesor a 256KB flash paměť. Obsahuje také tři výstupy, do kterých je možno zapojit tři servomotory a čtyři vstupy, pomocí kterých je možno zapojení senzorů viz Obrázek 1. Servomotory a senzory se zapojují pomocí kabelů typu RJ12. Dále chytrá programovatelná

kostka obsahuje jeden vstup pro USB, kterým lze propojit robota s počítačem, anebo je také možné propojení pomocí bezdrátového spojení bluetooth. Ovládacími prvky kostky jsou čtyři tlačítka. Dvě tlačítka pro pohyb v menu (doprava, doleva) a dvě tlačítka pro potvrzení,či návrat (enter a escape). Protože je stavebnice určena dětem obsahuje kostka také malý monochromatický maticový displej k zobrazení dat. Dalším prvkem je zabudovaný reproduktor k přehrávání zvuků. Programovatelnou kostku můžeme nabíjet dvěmi způsoby. Buďto adaptérem, součástí základní sady je i dobíjecí akumulátor, tedy baterie a nebo můžeme použít šest tužkových AA baterií.

Obrázek 1 Programovatelná kostka s příslušenstvím [JAKEŠ]

1.2 Servomotory

Servomotory jsou výstupním zařízení a také pohonem celého robota. Servomotor může sloužit také jako vstupní zařízení. K jedné inteligentní kostce lze připojit až tři servomotory, což je i standardní počet v základní stavebnici. Servomotory můžeme řídit samostatně, nezávisle na sobě, nebo spřaženě. Jeden servomotor má svůj rotační senzor, který měří otáčení motoru ve stupních, a to s přesností plus mínus jeden stupeň, ale také v přesnosti celých otáček. Jedno otočení odpovídá 360°. Rotační senzor je vestavěný a umožňuje nám přesné ovládání robota. Servomotor je možné nastavit na čtyři stupně rotace a to neomezeně, stupně, rotace a sekundy.

Neomezeně znamená, že se servomotor bude točit až do ukončení programu. V nastavení stupně se servomotor otáčí o předem daný počet stupňů. Rotace znamená, že se servomotor má předem dán počet rotací, které budou vykonány. No a sekundy, je udán počet sekund, ve kterých má servomotor rotovat. Servomotory mají své praktické využití. Můžeme je použít jako pohon našich robotů, ale můžeme je požít také k přemístění robota, či předmětů z jednoho místa na druhé, nebo pro zvedání, například u jeřábů.

Obrázek 2 Servomotor [JAKEŠ]

1.3 Dotykový senzor

Dotykový senzor je umístěn ve standardním krytu a to stejně jako ostatní NXT senzory, proto aby se daly vzájemně vyměňovat na určitém místě. Senzor se zapojuje standardně kabelem do NXT portu. Dotykový senzor funguje jako prvek vracející logické hodnoty pravda a nepravda (true a false). Tento senzor pracuje se třemi druhy akcí. Zmáčknutí (pressed), uvolnění (released) a zmáčknutí a opětovné uvolnění (bumped). Dotykový senzor má velkou škálu použití. Může sloužit jako senzor doteku, například při nárazu do překážky, nebo při zjištění přítomnosti překážky anebo jako koncové čidlo. Senzor může také sloužit jako spouštění a ovládání programu. Programu zatáčení do stran, vlevo a vpravo, nebo pro pohyby nahoru a dolů ale i další. Plochu dotykového senzoru je možné zvětšit připojením další plošky, neboť je senzor vybaven klasickým LEGO křížem, kde se dá připojit křížová tyčka a následně ploška z lega.

Obrázek 3 Dotykový senzor [JAKEŠ]

1.4 Světelný senzor

Světelný senzor umožňuje LEGO robotu vidět. Tento senzor umožňuje robotu rozlišovat světlo a tmu, ale i měření intenzity odraženého světla a to umožňuje robotu měřit intenzitu světla v místnosti a také rozpoznávání různých povrchů. Světlo, jeho intenzita a odraz je měřen, nebo spíše načítán v procentech. Světelný senzor je schopen porovnání vnější intenzity světla s vlastním procentuálním nastavením, k tomu slouží klasická znaménka větší než (>) a menší než (<). Senzor světla je také umístěn standardně v krytu, jako ostatní NXT senzory stavebnice. Samozřejmě se

dají senzory zaměňovat na určitých místech. Senzor se zapojuje klasickým kabelem do jednoho ze čtyř vstupních portů na inteligentní NXT kostce. Praktické využití je právě rozpoznávání intenzity světla i odraženého. Více se ovšem používá k rozpoznání barvy povrchu. Příkladem může být už několikrát zpracovaná úloha jízdy po čáře. Tato úloha se využívá i v praktickém životě, například v automobilkách kdy mají zásobovací roboti nakresleny na zemi vodící čáry, po kterých se pohybují, aby trefili na určené místo.

Obrázek 4 Světelný senzor [JAKEŠ]

1.5 Ultrazvukový senzor

Ultrazvukový senzor je standardní modul stavebnice. Tento senzor podobně jako senzor světelný umožňuje sestavenému robotu vidět. Ale nejen vidět. Díky jako kdyby očím, kterými je senzor opatřen, se může robot vyhýbat překážkám, hledat předměty, ale také zaznamenávat pohyb, či měřit vzdálenosti. Ultrazvukový senzor měří vzdálenost buďto v palcích, či centimetrech, do vzdálenosti až dvě stě padesát pět centimetrů, s přesností plus mínus tři centimetry. Senzor využívá podobný princip orientace jako netopýři. Ultrazvukový senzor vyšle akustickou vlnu, která má velmi vysokou frekvenci a proto je pro lidské ucho neslyšitelná a poté počítá dobu, za kterou se mu odražená vlna vrátí zpátky. Z této měřené doby si poté dopočte vzdálenost. Je to podobný princip jako u známé ozvěny. Praktické využití ultrazvukového senzoru je, že díky těmto očím dokáže robot například rozpoznání předem neočekávaných překážek. Příkladem z praxe života může být třeba automatický vysavač, který nezná prostředí a v okamžiku objevení nové neznámé překážky ji díky ultrazvukovému senzoru může objet, nebo před ní zastavit, anebo si také spočítá vzdálenost překážky.

Obrázek 5 Ultrazvukový senzor [JAKEŠ]

1.6 Zvukový senzor

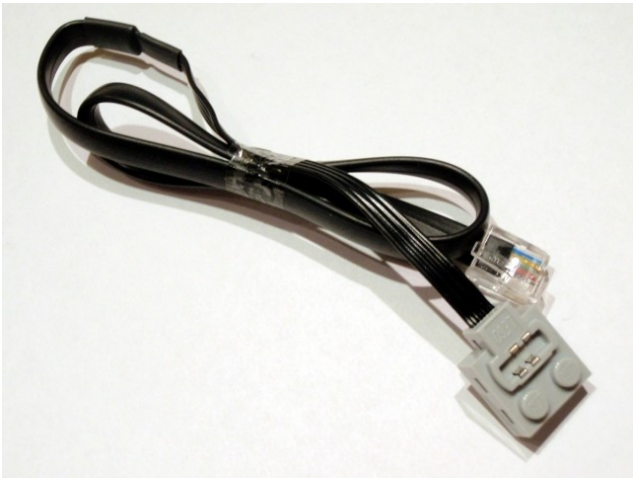
Zvukový senzor je v podstatě mikrofonní modul, který funguje jako detektor intenzity zvuku. Senzor měří intenzitu zvuku v decibelech (dB) a poskytuje také procentuální zobrazení hlasitosti (dBA). Zvukový senzor zachycuje a zobrazuje zvuky hlasité, ale taky zvuky velice tiché, zvuky které se blíží až k šumu místností. Naměřené hodnoty zvuku a hlasitosti, může senzor porovnávat s vlastním procentuálním nastavením a to pomocí znamének větší

(>) a menší (<). Senzor pracuje ve dvou režimech. Prvním je režim dB, kde se vyjadřuje hlasitost všech snímaných zvuků včetně zvuků, které mají velmi vysokou frekvenci a jsou pro člověka a jeho sluchové ústrojí nezachytitelné v decibelech. Druhým režimem je režim dBA, ve kterém se zobrazuje hlasitost v procentech, hlasitost slyšitelná lidským uchem. Zvukový senzor měří totiž akustický tlak do úrovně přibližně devadesát devět decibelů, to je sto procent. Tento hluk, zvuk odpovídá asi hluku, který vydává sekačka na trávu v chodu. Praktické využití tohoto zvukového senzoru může být například u bezpečnostních moderních zařízení. U rozbité výlohy kdy vznikne zvuk. Používá se také u analýzy zvuku.

Obrázek 6 Zvukový senzor [JAKEŠ]

1.7 Spojovací kabely

Spojovací kabely jsou velice důležité pro práci se stavebnicí. Používají se k propojení inteligentní kostky se všemi moduly, senzory a motory. Kabely se vyrábějí ve třech délkách a to 20, 35, 50 cm. V základní stavebnici jich je sedm a další se dají samozřejmě dokoupit.



Obrázek 7 Spojovací kabel [JAKEŠ]

1.8 Gyroskopický senzor

Gyroskopický senzor vyrábí společnost HiTechnic. Tento senzor slouží ke snímání směru natočení a rychlosti v jedné ose. Snímá v jednotkách stupeň za vteřinu. Maximum měření je 360°za sekundu. Použití pro měření rychlosti, ale také pro úlohy kde potřebujeme robota nějakým způsobem vybalancovat. Frekvence snímání je asi 300Hz, využívá analogový vstup.

Obrázek 8 Gyroskopický senzor [JAKEŠ]

1.9 Nová generace LEGO

V lednu představila společnost LEGO® novou generaci robotiky MINDSTORMS® EV3. Vzdělávací verze bude na trh uvedena pod názvem LEGO MINDSTORMS Education EV3. Nová generace LEGO má nový červeno šedý vzhled. V této podkapitole bych rád zmínil doplňkové motory a pohony LEGA nové generace, které se již dají objednat. Koupit však bude možné tyto díly až v září roku 2013. Novinkou v motorech a pohonech je EV3 Velký servomotor a Střední servomotor.

Obrázek 9 LEGO EV3

1.9.1 EV3 Velký servomotor

EV3 velký servomotor je výkonná pohonná jednotka využívající zpětnou vazbu z interního rotačního senzoru, což uživateli umožňuje ovládat otáčky motoru s přesností na jeden stupeň. Díky vestavěnému senzoru je možné programovat více motorů tak, aby se model pohyboval po zvolené trajektorii stejnou rychlostí. Servomotor obsahuje vlastní převodový mechanizmus.

Obrázek 10 EV3 Velký servomotor [The LEGO Group]

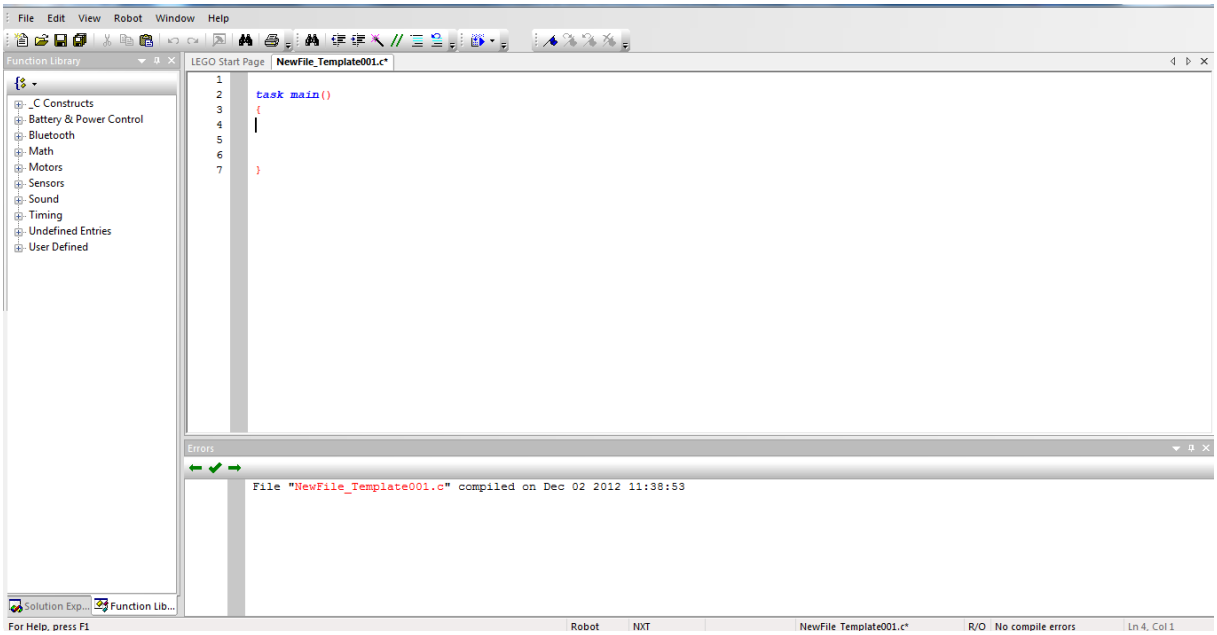
1.9.2 Střední servomotor

EV3 střední servomotor je vhodný do modelů s menší zátěží, požadavkem na vyšší rychlost a rychlou odezvu. Motor je opatřen interním rotačním senzorem, který umožňuje jeho ovládání s přesností jednoho stupně.

Obrázek 11 Střední servomotor [The LEGO Group]

2 RobotC

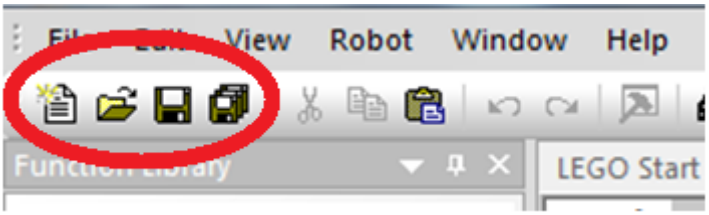
RobotC je velice silným programovacím nástrojem určený pro prostředí windows. Tento nástroj LEGA Mindstorms je celý založený na programovacím jazyku C, i když to není čistě C. V tomto prostředí, můžeme programovat náročnější úkoly a je možno řídit modely. Studenti, kteří pracují s tímto prostředím, se seznámí se zajímavým programovacím jazykem C, který jistě využijí v následném vzdělání, či v praxi. ROBOTC byl vyvinutý odborníky z Carnegie Mellon University's Robotics Academy, je určen pro práci s LEGO MINDSTORMS® NXT i RCX.



Obrázek 12 Programovací prostředí ROBOTC

2.1 Základní tlačítka

V pravém horním rohu se nachází ikonka pro otevření nového souboru. Nový soubor vypadá stejně jako okna na obrázku 10. Dále potom tlačítko pro otevření již uložených souborů, tlačítka pro uložení.

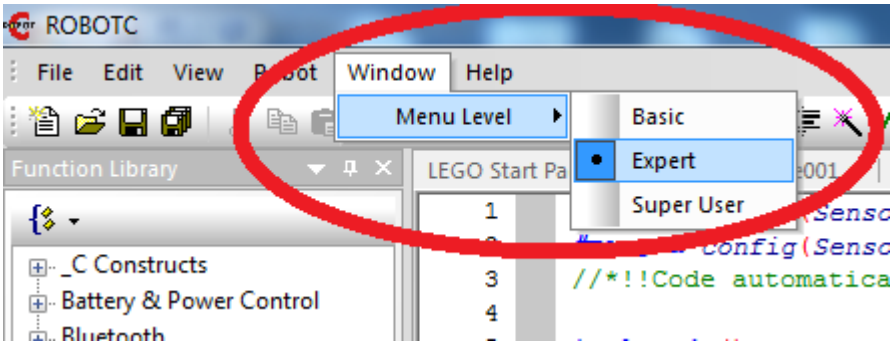


Obrázek 13 Základní tlačítka

-  Tlačítko pro otevření nového, prázdného souboru.
-  Tlačítko pro otevření již uloženého souboru.
-  Tlačítko pro uložení rozpracovaného souboru.
-  Tlačítko pro uložení všech otevřených, rozpracovaných souborů.

2.2 Nastavení obtížnosti

Na začátku práce si můžeme zvolit úroveň obtížnosti programu, jaké funkce nám budou zobrazeny a co všechno budeme moci použít. Podle nastavení obtížnosti nám prostředí RobotC bude nabízet množství využitelných funkcí a možností.

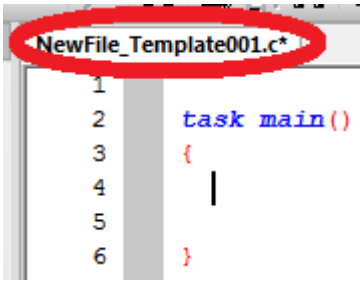


Obrázek 14 Záložka obtížnosti

Záložka **Window** v ní **Menu Level** a zde si podle obrázku nastavíme požadovanou obtížnost. Je možno vybírat ze tří, **Basic, Expert** a **Super User.**

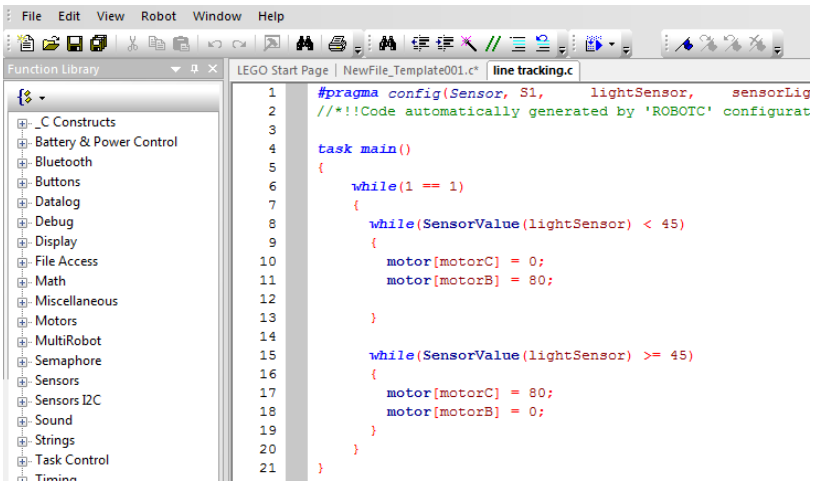
2.3 Okno pro vlastní program

Uprostřed zobrazení je největší okno, do kterého píšeme vlastní program nazvaný prostředím defaultně ***Newfile_Template001.c****, samozřejmě je možné ho přejmenovat při ukládání.



Hvězdička ***.c****, znamená, že je soubor ještě neuložen. Podle této malé hvězdičky poznáme, jestli jsme soubor už uložili.

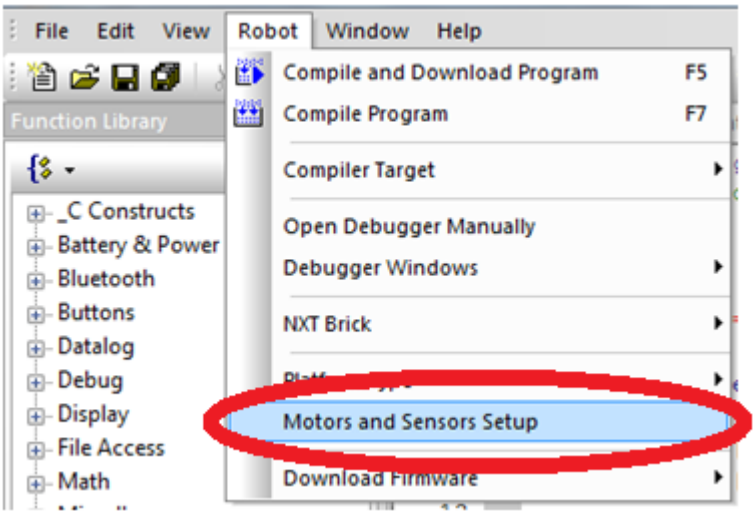
Každý program musí začínat úvodní funkcí, v tomto případě je to task main (). Protože je funkce main bezrozměrná a nemá žádnou hodnotu, jsou závorky za ní prázdné, viz programovací jazyk C. Každý řádek musí být oddělen středníkem. Svůj vlastní program vepisujeme mezi složené závorky - { a }. Program si sám kód vede, odsazuje po stlačení klávesy enter. Je však možno si odsazení a řádkování upravovat sami, mezerami, odstavci apod.



Obrázek 15 Vlastní program

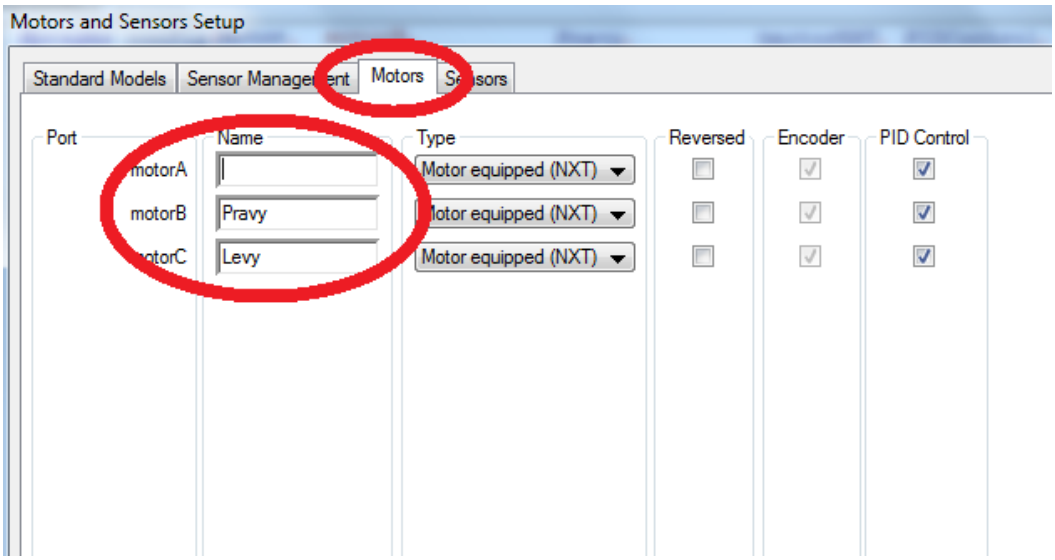
2.4 Nastavení motorů

Při práci si musíme nastavit motory a senzory. Záložka **Robot** a následně **Motor and Sensor Setup**. Zde si motory a senzory pojmenujeme, vytvoříme si jakoby vlastní knihovnu.



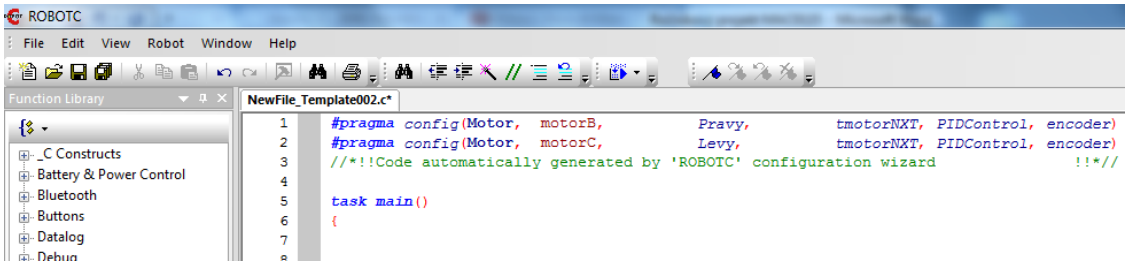
Obrázek 16 Motory a senzory

Samotné nastavení motorů je velice jednoduché. **Name** – pojmenujeme si podle sebe motory, senzory.



Obrázek 17 Natavení motorů

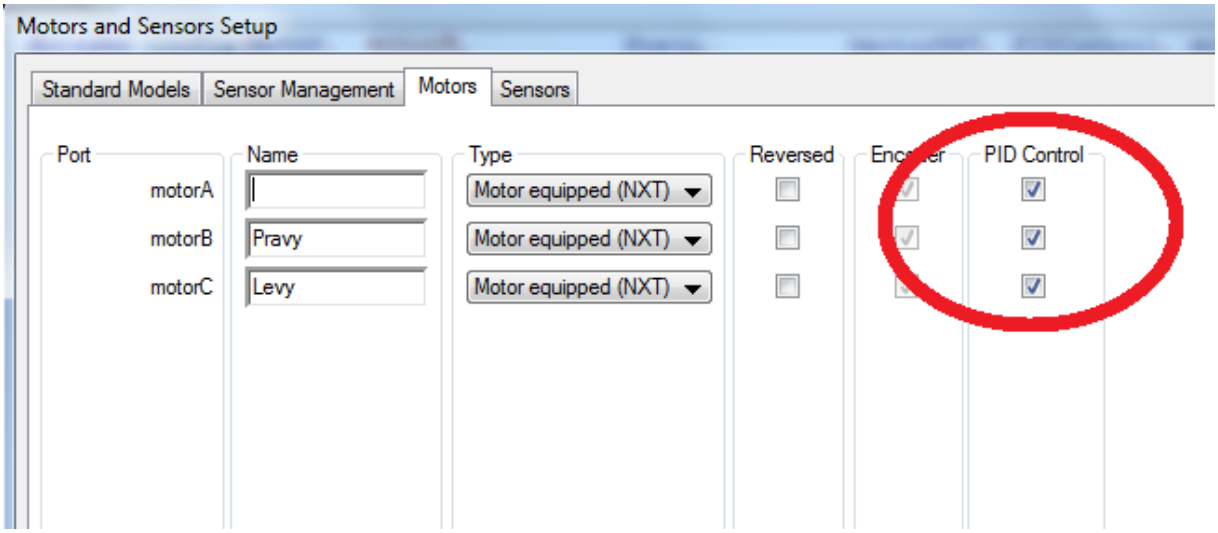
V záložce **Motors** – motory si můžeme pojmenovat vlastním jménem. Nebudeme muset poté pokaždé do našeho programu vypisovat např. motor[motorA], ale bude nám stačit si napsat pouze naše pojmenování motoru, třeba jen A, nebo pravý motor, levý motor, abychom se v kódu dobře orientovali. V programu se nám ukáže nastavení motorů.



Obrázek 18 Nastavené motory

2.5 Nastavení PID regulace motorů

Další zajímavostí je PID regulace, která je v prostředí RobotC přímo funkcí. Najedeme no nastavení motorů a senzorů – záložka **Robot** – **Motors and Sensors Setup**. Pokud si zaškrtneme políčka **PID Control**, můžeme potom v programu zapínat, nebo vypínat PID regulaci rychlosti.



Obrázek 19 nastavení PID regulace

Tato regulace se používá při udržování konstantní rychlosti, bez ohledu na povrch a odpor. Pro vyvolání této regulace motorů se v programu používá funkce nMotorPIDSpeedCtrl[.....], která má více modifikací.

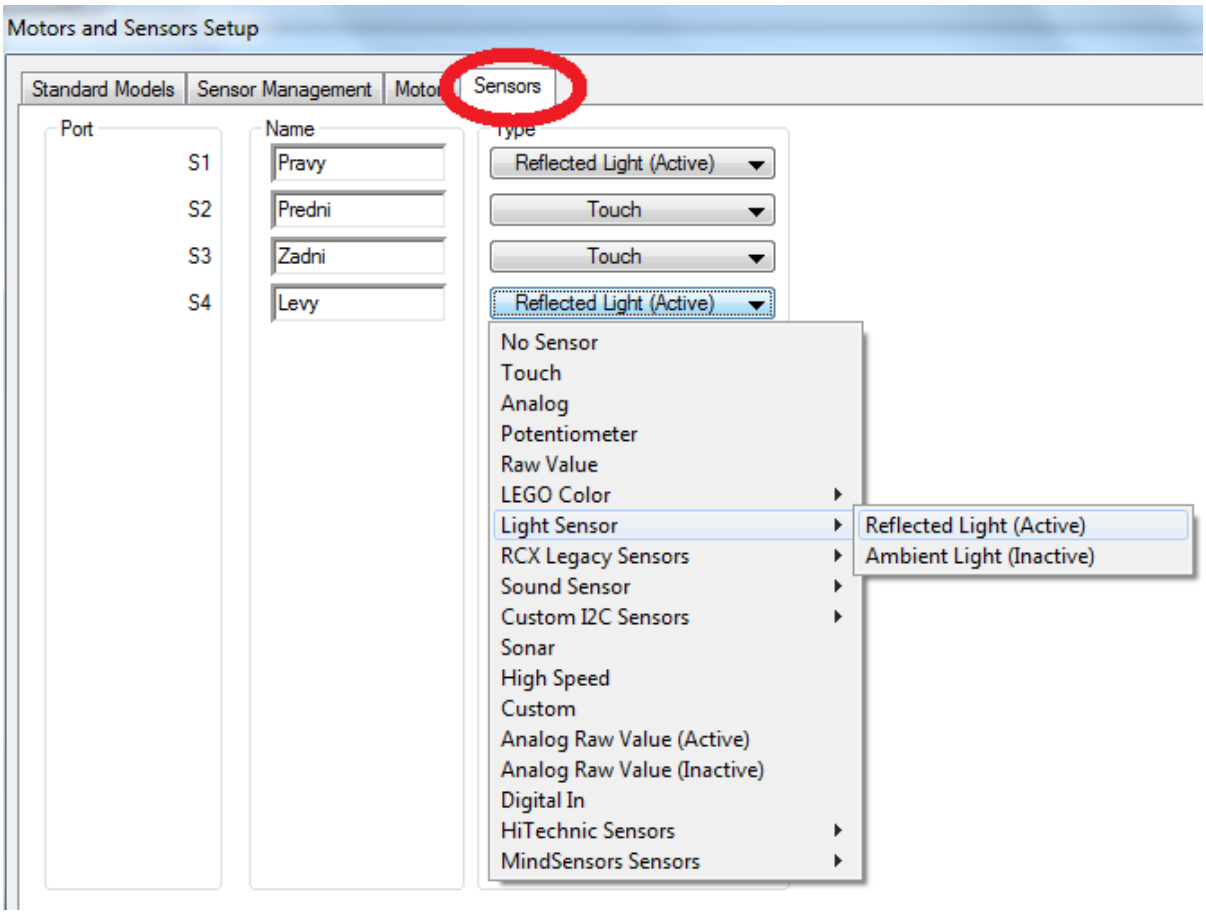
```
nMotorPIDSpeedCtrl[motorB] = mtrSpeedReg;
nMotorPIDSpeedCtrl[motorC] = mtrSpeedReg;

  nMotorEncoder[motorC] = 0;
  nMotorEncoder[motorB] = 0;
```

Obrázek 20 Regulace motorů

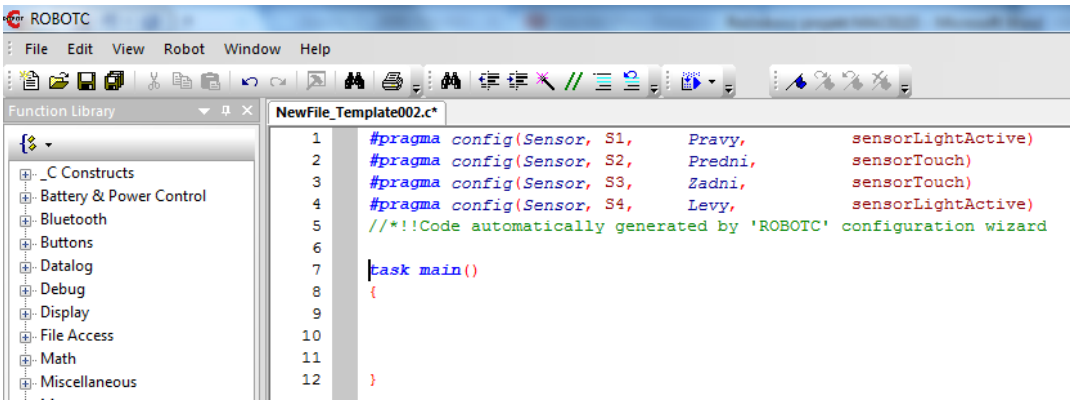
2.6 Nastavení senzorů

Nastavení senzorů není také nic těžkého. Provádí se podobně, také na záložce **Robot - Motors and Sensors Setup - Sensors**, kde si je možné vybrat z mnoha senzorů.



Obrázek 21 Nastavení senzorů

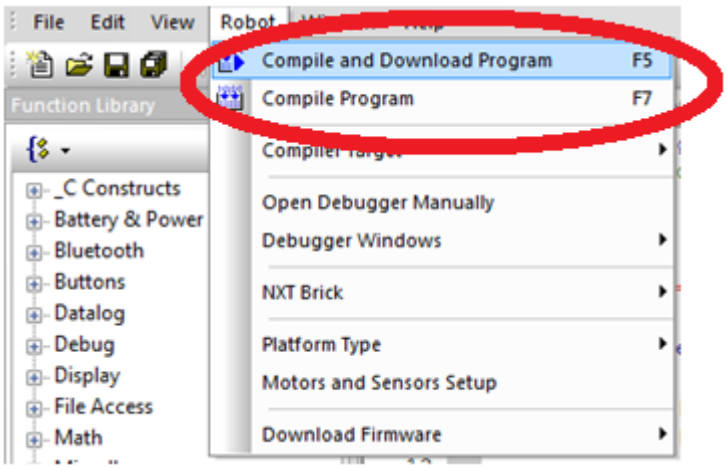
Senzory si stejně jako motory pojmenujeme vlastními jmény a poté jen podle těchto jmen voláme do programu. Protože máme čtyři porty, je možnost použití čtyř senzorů a máme možnost si vybrat z nespočetného množství senzorů.



Obrázek 22 Nastavené senzory

2.7 Kompilace programu

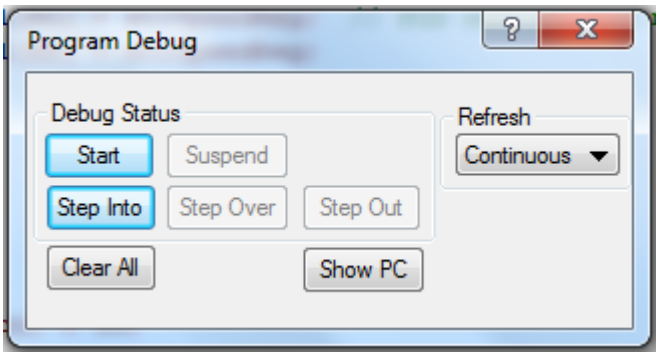
Důležitou záložkou pro práci s RobotC a vlastním robotem, je záložka Robot. Kde si příkazem **Compile Program**, zkratkou **F7** ověříme správnost programu, nebo záložkou **Compile and Download Program**, zkratkou **F5** náš napsaný program uložíme a pokud máme připojeného robota přes USB kabel a zapnutou NXT kostku, uložíme program do robota. Je možné si i vyzkoušet jak se chová. Případné chyby v gramatice jazyka C, lze jednoduše přepsat, smazat, opravit.



Obrázek 23 Záložka Robot

2.8 Test programu

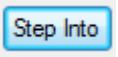
Po spuštění kompilace, zmáčknutí tlačítka **F5** nám, pokud je robot připojen k počítači přes USB kabel, vyskočí tato tabulka.



Obrázek 24 Test programu

Nabízí se nám několik tlačítek, pro test programu.

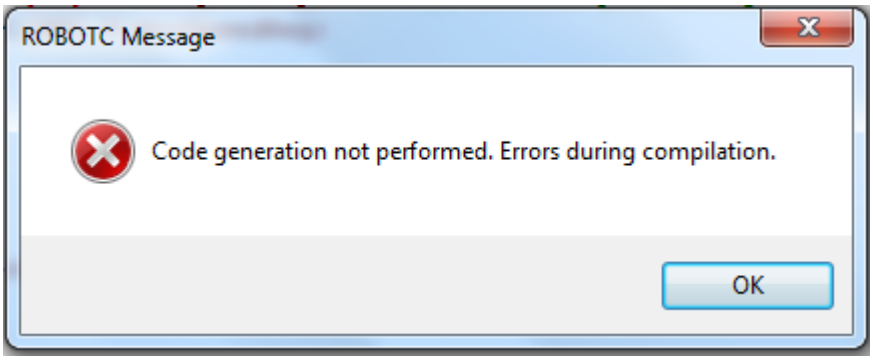
Tlačítko **Start** pro spuštění testu programu, kdy robot připojený kabel k PC začne konat napsané funkce.



Tlačítko **Step Into**. Toto tlačítko nám umožní projet si celý program krok po kroku. Vidíme tak, jak se robot chová v jednotlivých úsecích kódu.

2.9 Chyby programu

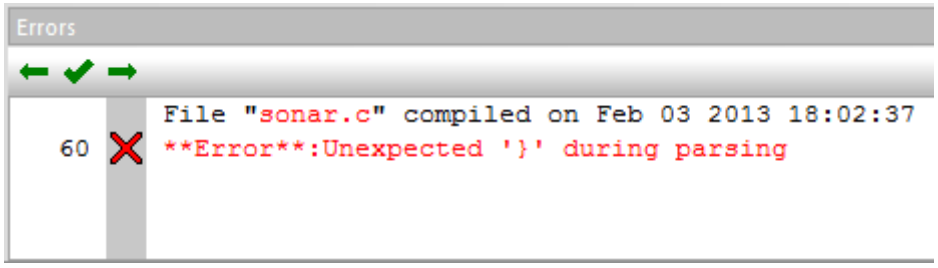
Pokud chceme program kompilovat a tedy uložit do našeho robota a je chybný, vyskočí nám tato tabulka, která nás upozorňuje, že něco není v pořádku.



Obrázek 25 Hlášení o chybě

Zároveň se nám také v okně nazvaném Errors – chyby, poruchy, zobrazí hlášení o tom, co v daném programu chybí a upozorní nás také, na kterém řádku našeho programu došlo k pochybení.

Na obrázku vidíme, že je chyba na šedesátém řádku a že chybou je chybějící závorka – ‘ } ‘. Chybu opravíme a kompilujeme opětovně.



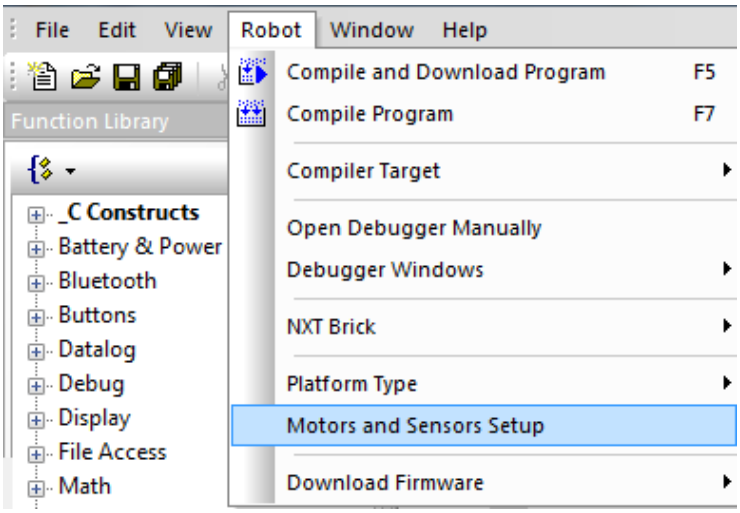
Obrázek 26 Okno pro chyby

3 Demonstrativní úlohy

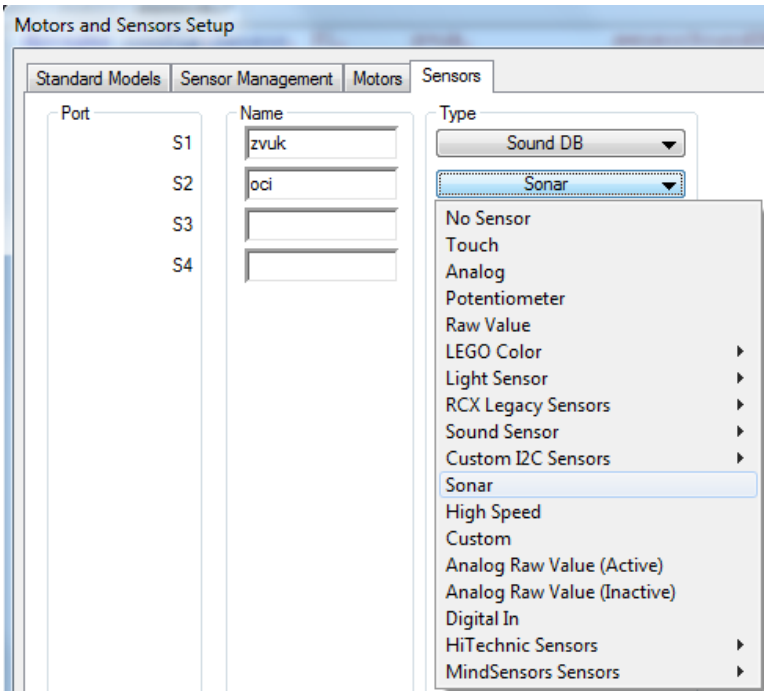
Pro realizaci demonstrativních úloh jsou vybrány tyto, primitivní hlídací robot, kalibrace senzorů, jízda po čáře, úloha průjezdu univerzálním bludištěm a hlasem dálkově ovládaný robot.

3.1 Hlídací robot

Hlídací robot je velice jednoduchá úloha za použití ultrazvukového senzoru a senzoru zvukového. Robot je například u okna, nebo dveří na svém hlídacím stanovišti. Pokud bude něco v nepořádku, jestliže se například otevřou dveře, ultrazvukovému senzoru se změní vzdálenost od překážky, začne robot vyvíjet alarm v podobě rychlého rotování a zvukového signálu. V nabídce **Robot** → **Motors and Sensors Setup** si nastavíme motory a senzory.



Obrázek 27 Záložka Robot



Obrázek 28 Nastavení senzorů

Úplně stejně jako v bodu 2.

```
#pragma config(Sensor, S1,     zvuk,           sensorSoundDB)
#pragma config(Sensor, S2,     oci,            sensorSONAR)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
```

Obrázek 29 Nastavené motory a senzory

Program musí začínat úvodní funkcí a smyčkou.

```
task main()
{
  while ( 1 )
  {
    ....
  }
}
```

Obrázek 30 Úvodní funkce a cyklus

Pokud je vzdálenost ultrazvukového senzoru menší než nastavených 30 jednotek, potom se robot, dle zadání roztočí a vyvíjí zvukový alarm. Tímto nás upozorní, že je něco v nepořádku. Roztočení docílíme tak, že jeden motor, [motorB], je nastaven na 70 ot/min a druhý motor, [motorC], by mohl být

nastaven na 0 ot/min, ale z důvodu rychlejšího otáčení a otáčení na místě je motor nastaven na hodnotu -70 ot/min.

```
if (SensorValue(oci) < 30)
{
  motor[motorB] = 70;
  motor[motorC] = -70;
  PlaySound(soundBeepBeep);
}
```

Obrázek 31 Robot koná pohyb a zvukový alarm

3.2 Kalibrace světelných senzorů

Kalibrace „na tlačítko“. Pro kalibraci světelných senzorů musíme nejprve najít hodnotu „černé“ a hodnotu „bílé“, pro světlo. Program začneme kalibračním režimem.

```
int leftValue = 0;
int rightValue = 0;
```

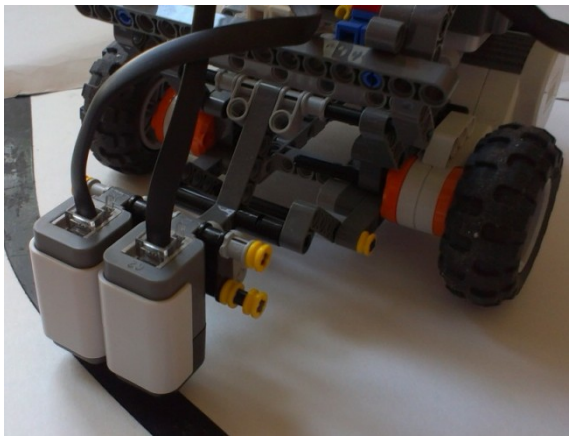
Obrázek 32 Hodnoty pro senzory

Přednastavíme si hodnotu levého i pravého světelného senzoru jako nula. To proto, aby si robot, následným zápisem nastavil hodnotu černé a bílé sám, aby mohl provádět čtení čáry v jakémkoliv prostředí, například v šeru, nebo naopak při velmi jasném světle, denním, či umělém.

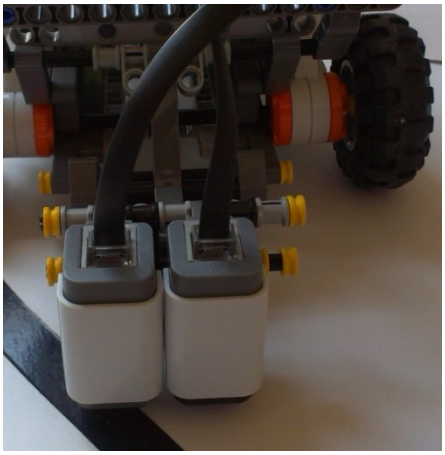
```
int leftLow = 1000; //Velke cislo - nejtmavejsi
int rightLow = 1000;
int leftHigh = 0; // Male cislo - nejsvetlejsi
int rightHigh = 0;
```

Obrázek 33 Nejtmavější a nejsvětlejší hodnoty

Pro ulehčení práci si vlastně uložíme nejvyšší a nejnižší hodnoty, černé a bílé, které načítají oba senzory, jak pravý tak i levý senzor. Musíme se ujistit, že je robot při kalibraci alespoň jednou v nejtemnějším místě a alespoň jednou v místě nejsvětlejším, jako na obrázcích.



Obrázek 34 Nejtmavější místo



Obrázek 35 Nejsvětlejší místo

Načítání nejtmavější hodnoty.

```
//Nacitani vysoke a nizke hodnoty pro levy senzor
if(leftValue > leftHigh)
  {
  leftHigh = leftValue;
   }
else if(leftValue < leftLow)
  {
  leftLow = leftValue;
  }
```

Obrázek 36 Načítaní nejvyšší a nejnižší hodnoty levého senzoru

Načítání a ukládání hodnoty nejnižší, nejsvětlejší.

```
//Nacitani vysoke a nizke hodnoty pro pravy senzor
if(rightValue > rightHigh)
  {
  rightHigh = rightValue;
  }
else if(rightValue < rightLow)
  {
    rightLow = rightValue;
  }
```

Obrázek 37 Načítání nejvyšší a nejnižší hodnoty pravého senzoru

3.2 Jízda po čáře

V této podkapitole si popíšeme program jízdy po čáře. Na začátku práce si otevřeme prázdný dokument a nastavíme motory a senzory, které budeme používat pro naši úlohu. Nastavení provedeme stejně jako v první úloze, či podle postupu nastavení dle v bodu 2. Jízdu po čáře uskutečníme pomocí dvou světelných senzorů, které si oba nastavíme jako **sensorLightActive**.
Motory si nastavíme s PID regulací, abychom jsme se mohli pohybovat rychleji bez ohledu na povrch a odpory.

```
#pragma config(Sensor, S1,     pravy,          sensorLightActive)
#pragma config(Sensor, S2,     levy,           sensorLightActive)
#pragma config(Motor,  motorB,          B,             tmotorNXT, PIDControl, encoder)
#pragma config(Motor,  motorC,          C,             tmotorNXT, PIDControl, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
```

Obrázek 38 Motory a senzory

Jako každý program, tak i tento musí začínat úvodní fukncí.

```
task main()
{
  ...
}
```

Obrázek 39 Úvodní funkce

Pomocí funkce nMotorPIDSpeedCtrl zapneme regulaci motorů B i C. Funkce nMotorEncoder vynuluje motory po uplynutí přibližně 90 otáček.

```
nMotorPIDSpeedCtrl[motorB] = mtrSpeedReg;
nMotorPIDSpeedCtrl[motorC] = mtrSpeedReg;
```

Obrázek 40 vyvolání regulace motorů

Cyklus while (1) nám uzavírá program do smyčky. Pokud bude hodnota v závorce kladná, spustí se program.

Nastavíme si funkci float, která vrací absolutní hodnotu čísla. Pokud tedy použijeme v programu písmeno L, bude nám načtena absolutní hodnota levého světelného senzoru. Pokud R, tak hodnota pravého senzoru, podle nastavení

```
float L = SensorValue(levy);
float R = SensorValue(pravy);
```

Obrázek 41 Float

Pokud bude hodnota levého senzoru menší než 45, levý, tedy motorC se vypne a motorB pojede dále rychlostí 60ti otáček.

Funkce nxtDisplayCenteredTextLine nám bude vypisovat hodnoty ze senzorů na displej NXT kostky.

```
if(L < 45)
{
     nxtDisplayCenteredTextLine(1, "SensorLt50: %d", SensorValue[levy]);
     nxtDisplayCenteredTextLine(6, "SensorRt: %d", SensorValue[pravy]);

    motor[motorC] = 0;
    motor[motorB] = 60;
}
```

Obrázek 42 float pro levý senzor

Pokud bude hodnota pravého senzoru větší než 45, tak se vypne pravý motor, tedy motorB. Hodnoty se nám zase vypíší na displej programovatelné kostky.

```
else if(R > 45)
{
   nxtDisplayCenteredTextLine(1, "SensorLt50: %d", SensorValue[levy]);
   nxtDisplayCenteredTextLine(6, "SensorRt: %d", SensorValue[pravy]);

  motor[motorC] = 60;
  motor[motorB] = 0;
}
```

Obrázek 43 float pro pravý senzor

V případě, že nedojde ani k jedné z akcí pod smyčkou while, znamená to, že robot nevidí čáru, nemá kontrast mezi bílou a černou a nevidí nic. Hodnoty pravého ani levého senzoru nejsou ani větší ani menší než 45, a proto se zastaví.

```
motor[motorC] = 0;
motor[motorB] = 0;
```

Obrázek 44 Nulované motory

Oba motory se vypnou.

3.4 Průjezd bludištěm

V této podkapitole si podrobně popíšeme a vysvětlíme dílčí úlohy pro úspěšné zpracování průjezdu univerzálním bludištěm. Na začátku zpracování této úlohy si po otevření nového souboru nastavíme motory a senzory pro jednodušší práci.

```
#pragma config(Motor,  motorB,           ML,            tmotorNXT, PIDControl, encoder)
#pragma config(Motor,  motorC,           MP,            tmotorNXT, PIDControl, encoder)
```

Obrázek 45 Nastavení motorů

Motory jsou pojmenovány podle toho, na které straně motor je, tedy ML – levý motor, MP – pravý motor.

```
#pragma config(Sensor, S1,     oci,           sensorSONAR)
#pragma config(Sensor, S2,     predek,        sensorTouch)
```

Obrázek 46 Nastavení senzorů

Pro průjezd bludištěm je zapotřebí senzor ultrazvukový a dotykový senzor, který je umístěn vepředu robota. Dotykový senzor je pojmenován podle toho, ve které části robota se nacházejí.

3.4.1 Robot čte zeď, otočka vpravo

V této podúloze robot jede podél zdi a to tak, že načítá ultrazvukovým senzorem, umístěným na levé straně robota, vzdálenost, v jaké se překážka, zeď nachází. Pokud robot vidí, načítá zeď, tak jede rovně konstantní rychlostí, to znamená, že jsou motory nastaveny na stejnou hodnotu otáček.

```
task main()
{
  wait1Msec(2000);
  while ( 1 )
  {
    if (SensorValue(oci) < 30) // Vidi stenu jede rovne podel ni
    {
      motor[ML] = 40;
      motor[MP] = 40;
    }
    .....
  }
}
```

Obrázek 47 Načítání vzdálenosti

Pokud ultrazvukový senzor načte vzdálenost větší, než která je nastavena zatočí vpravo o devadesát stupňů.

```
else
{
  if (SensorValue(oci) > 30) // Nevidi stenu zataci
  {
            nMotorEncoder[ML] = 0;
    while (nMotorEncoder[ML] < 360)
        {
          motor[MP] = -20;
          motor[ML] = 80;
        }
  }
}
```

Obrázek 48 Otočka vpravo

Funkce enMotorEncoder vynuluje pravý motor a pokud ještě otočka nedosáhla 360° počítá, tak se levý motor nuluje a pravý motor plně akceleruje. To nám

zajistí vyjet pravoúhlou zatáčku. Poté zase robot načte zeď a pokračuje dále průjezdem bludiště.

3.4.2 Robot couvá, otočka vpravo

Dále v programu potřebujeme otočku vpravo, anebo když se robot dostane do slepé uličky, potřebujeme, aby vycouval. Pokud robot čte zeď po pravé straně, a bludiště vede doleva robot, jede, až narazí do zdi. Při stlačení předního dotykového senzoru robot zastaví, začne couvat a poté zatočí doprava.

```
if (SensorValue(predek) == 1) // Kdyz narazi
  {
    while (time1[T1] < (3000))
    {
      motor[ML] = -20;          // Couva
      motor[MP] = -20;

      wait1Msec(2000);
    }
    .......
  }
```

Obrázek 49 Stlačení dotykového senzoru

Robot couvá. Mínus před hodnotou motoru nám zajišťuje couvání. Poté robot zatočí doleva a počká, aby mohl načíst zeď bludiště a mohl pokračovat dále průjezdem.

```
nMotorEncoder[MP] = 0;
while (nMotorEncoder[MP] < 360)
{
  motor[MP] = 80;
  motor[ML] = -20;
}
```

Obrázek 50 Otočka doleva

Funkce nMotorEncoder tentokrát nuluje pravý motor, přesně naopak, než při otočce doprava, to znamená, že levý motor zůstane vypnutý a pravý motor plně akceleruje, zatočí vpravo.

3.5 Dálkově zvukem řízený robot

Dálkově řízený robot pomocí zvuku je nová úloha na naší škole ještě nezpracovaná. Problematika této úlohy spočívá v intenzitě, tedy v hlasitosti vydávaného zvuku, podle kterého robot jede buďto doprava, nebo doleva.

K této úloze je využit již sestavený robot se dvěma motory a zadním otočným kolečkem, zvukový senzor a přístroj k vydávání zvuku. Který je umístěn na vrcholu kabiny. Na začátku práce si musíme zase nastavit motory a senzor, pro snadnější práci, stejně jako v předchozích úlohách, či bodu 2.

```
#pragma config(Sensor, S1,     zvuk,           sensorSoundDBA)
#pragma config(Motor,  motorB,          Levy,          tmotorNXT, PIDControl, encoder)
#pragma config(Motor,  motorC,          Pravy,         tmotorNXT, PIDControl, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
```

Obrázek 51 Nastavení motorů a senzoru

Program začíná už klasicky úvodní funkcí a dvou sekundovým čekáním. Pro přípravu, aby robot nevyrazil vpřed hned po zmáčknutí tlačítka start.

```
task main()
{
  wait1Msec(2000);
  ....
}
```

Obrázek 52 Úvodní funkce a čekání

Po spuštění smyčky while (1) a potom, co robot počká dvě sekundy začne konat pohyb přímočarý, jede rovně, konstantní rychlostí 50ot/min, oba motory jedou na stejný výkon. Protože nastavená hodnota zvukového senzoru je menší než 55dB, tzn. „ticho“(žádné zvukové výkyvy).

```
while ( 1 )
{
  if (SensorValue(zvuk) < 55)
  {
    motor[Levy] = 50;
    motor[Pravy] = 50;
  }
  ....
}
```

Obrázek 53 Přímočarý pohyb

Funkce else if znamená, že pokud se hodnota zvuku zvýší, dle zápisu bude větší než 55dB a zároveň ale menší než 65dB, je to znamení pro robota aby zatočil vpravo. Levý motor zůstane konstantní, tedy 50ot/min a pravý motor, pro rychlejší zatočení vykoná -20ot/min, pojede jakoby proti. To nám umožní větší rychlost zatočení.

```
else if ((SensorValue(zvuk) >  55) && (SensorValue(zvuk) < 65))
{
  motor[Levy] = 50;           // Zataci vpravo
  motor[Pravy] = -20;
}
```

Obrázek 54 Zatočení vpravo, zvýšení zvuku

Pokud, else if, zvuk přesáhne hodnotu 65dB, potom robot zatočí vlevo. Podle toho jak dlouho zvuk nižší, nebo vyšší trvá, se robot otáčí, nebo jede zase rovně.

```
else if (SensorValue(zvuk) > 55)
{
  motor[Levy] = -20;       // Zataci vlevo
  motor[Pravy] = 50;
}
```

Obrázek 55 Zatočení vlevo

4 Závěr a zhodnocení dosažených výsledků

V prvním bodě jsem se seznámil se stavebnicí se všemi komponenty, které základní i rozšířená sada LEGA Mindstorms obsahuje. Seznámil jsem se také se všemi pohony, servomotory a s určitou částí senzorů. Motory i senzory byly popsány. Senzory, které jsou použity v demonstrativních úlohách, jsou popsány podrobněji a jsou vysvětleny i funkce pomocí kterých senzory používáme v námi navržených programech pro zpracování těchto úloh. V další části prvního bodu jsme se seznámili s nejnovějšími servomotory a komponentami LEGA Midstorms, takzvané řady EV3, které mají zcela nový vzhled i rozdílné funkce.

V druhém bodě jsem zpracoval jednoduchou příručku, pomocí které se dá rychle začít pracovat s prostředím robotC. Jsou zde vysvětlena základní tlačítka, nastavení motorů a senzorů, vysvětleny chybové hlášky a také kompilace a uložení programu do robota, ale také zkouška uloženého programu.

Ve třetím bodě jsou podrobně popsány demonstrativní úlohy. Jízda po čáře, kalibrace světelných senzorů, dílčí úlohy průjezdu bludištěm, čtení zdi ultrazvukovým senzore a zatáčení, couvání po nárazu pomocí dotykového senzoru umístěného v přední části robota. Poslední úlohou je nová úloha dálkově řízeného robota zvukem, kdy robot reaguje na intenzitu, síly zvuku. Směry dalšího řešení by mohlo být rozvedení, ztížení těchto úloh, případné spojování těchto úloh v jednu složitější a praktické využití sepsané příručky do běžné výuky.

Seznam použité literatury

DALE YOCUM [online]. *NXT Tutorial.* Dale Yocum. 2007. dostupné z WWW: <URL: http://www.ortop.org/NXT_Tutorial/>.

FARANA, R., SMUTNÝ, L., VÍTEČEK, A. 1999. *Zpracování odborných textů z oblasti automatizace a informatiky*. 1. vyd. Ostrava : VŠB-TU Ostrava, 1999. 68 s. ISBN 80-7078-737-6.

HEROUT, P. *Učebnice jazyka C 1.díl* 5.vyd. Praha : KOPP Praha, 2008. 280 str. - ISBN: 80-7232-351-2

KAČMÁŘ, D. *Jazyk C* 1. vyd. Praha, Computer Press 2000 , 185 str. ISBN: 80-7226-295-5

JAKEŠ, T. Robotické vzdělávání: LEGO Mindstorms. *Robotické vzdělávání* [online]. 2010 - 2012 [cit. 2013-02-01]. Dostupné z WWW:<URL: https://www.lego.zcu.cz/web/>

ROBOTC *Teaching ROBOTC for LEGO MINDSTORMS*. dostupné z WWW: <URL: http://www.robotc.net/>.

EDUXE: LEGO Education. EDUXE S.R.O., Velké Pavlovice. *Robotika EV3* [online]. 2012. vyd. [cit. 2013-05-02]. Dostupné z: <http://www.eduxe.cz/les/robotika-ev3/>

HiTechnics, *HiTechnics Products,* aktualizace 2010, [online], [cit. 2010], dostupné z WWW <URL:http://www.hitechnics.com/>

LEGO *LEGO® Education,* aktualizace 2010, [online], [cit. 2010], dostupné z WWW <URL:http://www.legoeducation.us/global.aspx>

PHExtreme NXT, *Philo´s Home Page,* aktualizace 2010, [online], [cit. 2010], dostupné z WWW <URL:http://philophone.com/>

Přílohy

V příloze jsou umístěny všechny programy. Program pro alarm, kalibraci, jízdu po čáře, průjezd bludištěm a zvukem ovládaného robota.

Příloha 1

Zdrojový program alarmu.

```
#pragma config(Sensor, S1,     zvuk,           sensorSoundDB)
#pragma config(Sensor, S2,     oci,            sensorSONAR)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

task main()
{
  while ( 1 )
  {
    if (SensorValue(oci) < 30)
    {
      motor[motorB] = 70;
      motor[motorC] = -70;
      PlaySound(soundBeepBeep);
    }
  }
}
```

Příloha 2

Zdrojový program pro kalibraci světelných senzorů.

```
#pragma config(Sensor, S1,     pravy,          sensorLightActive)
#pragma config(Sensor, S2,     levy,           sensorLightActive)
#pragma config(Sensor, S3,     Tlacitko,       sensorTouch)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

task main()
{
  int leftValue = 0;
  int rightValue = 0;

  int leftLow = 1000; //Velke cislo - nejtmavejsi
  int rightLow = 1000;
  int leftHigh = 0; // Male cislo - nejsvetlejsi
  int rightHigh = 0;

  //Auto-kalibrace ukoncime stiskem tlacitka. A robot zacne sledovat caru

  while(!SensorValue[Tlacitko])
    {
    leftValue = SensorValue[levy];
    rightValue = SensorValue[pravy];

    //Nacitani vysoke a nizke hodnoty pro levy senzor
    if(leftValue > leftHigh)
      {
      leftHigh = leftValue;
       }
    else if(leftValue < leftLow)
      {
      leftLow = leftValue;
      }


   //Nacitani vysoke a nizke hodnoty pro pravy senzor
   if(rightValue > rightHigh)
     {
     rightHigh = rightValue;
     }
     else if(rightValue < rightLow)
     {
       rightLow = rightValue;
     }
   }
}
```

Příloha 3

Zdrojový program pro jízdu po čáře.

```
#pragma config(Sensor, S1,     pravy,          sensorLightActive)
#pragma config(Sensor, S2,     levy,           sensorLightActive)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

task main()
{

    nMotorPIDSpeedCtrl[motorB] = mtrSpeedReg;  // PID regulace zapnuta pro motory
    nMotorPIDSpeedCtrl[motorC] = mtrSpeedReg;

      nMotorEncoder[motorC] = 0;
      nMotorEncoder[motorB] = 0;

    while( 1 )
    {
      float L = SensorValue(levy);
      float R = SensorValue(pravy);

    if(L < 45)                //  Jestlize je L mensi nez 45 levy motor se vypne
    {
         nxtDisplayCenteredTextLine(1, "SensorLt50: %d", SensorValue[levy]);  // ukaze na displeji hodnoty
         nxtDisplayCenteredTextLine(6, "SensorRt: %d", SensorValue[pravy]);

        motor[motorC] = 0;
        motor[motorB] = 60;
    }


    else if(R > 45)          // Jestlize je R vetsi nez 45 pravy motor se zapne a levy motor se vypne
    {
       nxtDisplayCenteredTextLine(1, "SensorLt50: %d", SensorValue[levy]);
       nxtDisplayCenteredTextLine(6, "SensorRt: %d", SensorValue[pravy]);

      motor[motorC] = 60;
      motor[motorB] = 0;
    }
  }
      motor[motorC] = 0;
      motor[motorB] = 0;
}
```

Příloha 4

Zdrojový program pro průjezd universálním bludištěm.

```
#pragma config(Sensor, S1,     oci,           sensorSONAR)
#pragma config(Sensor, S2,     predek,        sensorTouch)
#pragma config(Motor,  motorB,          ML,            tmotorNXT, PIDControl, encoder)
#pragma config(Motor,  motorC,          MP,            tmotorNXT, PIDControl, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

task main()
{
  wait1Msec(2000);
  while ( 1 )
  {
    if (SensorValue(oci) < 30) // Vidi stenu jede rovne podel ni
    {
      motor[ML] = 40;
      motor[MP] = 40;
    }
    else
    {
      if (SensorValue(oci) > 30) // Nevidi stenu zataci do prava
      {
                nMotorEncoder[ML] = 0;
        while (nMotorEncoder[ML] < 360)
            {
              motor[MP] = -20;
              motor[ML] = 80;
            }

    }
  }
    if (SensorValue(predek) == 1) // Kdyz narazi
      {
        while (time1[T1] < (3000))
        {
          motor[ML] = -20;          // Couva
          motor[MP] = -20;

          wait1Msec(2000);
        }

        nMotorEncoder[MP] = 0;
        while (nMotorEncoder[MP] < 360)
        {
          motor[MP] = 80;               // Zataci v levo
          motor[ML] = -20;
        }
      }
    }
```

Příloha 5

Zdrojový program pro zvukem, dálkově ovládaného robota.

```
#pragma config(Sensor, S1,     zvuk,           sensorSoundDBA)
#pragma config(Motor,  motorB,          Levy,          tmotorNXT, PIDControl, encoder)
#pragma config(Motor,  motorC,          Pravy,         tmotorNXT, PIDControl, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

task main()
{
  wait1Msec(2000);
  while ( 1 )
  {
    if (SensorValue(zvuk) < 55)
    {
      motor[Levy] = 50;
      motor[Pravy] = 50;
    }
    else if ((SensorValue(zvuk) >  55) && (SensorValue(zvuk) < 65))
    {
      motor[Levy] = 50;           // Zataci vpravo
      motor[Pravy] = -20;
    }
    else if (SensorValue(zvuk) > 55)
    {
      motor[Levy] = -20;      // Zataci vlevo
      motor[Pravy] = 50;
    }
  }
}
```